FPLA Mechanization of Arithmetic Elements to Produce A + B or to Pass A Only

D. E. Wallis, H. Taylor, and A. L. Rubin
Communications Systems Research Section

This article describes a 4-bit and a 3-bit adder which can be implemented under special hardware restrictions. The chip to be used is Field-Programmable Logic Array (FPLA) with 12 input lines, 50 AND gates inside, and output through only 6 OR gates. The context in which it is being used requires an "enable" function which can suppress one of the two numbers to be added. The 3-bit enabled adder is compatible with lookahead-carry mechanizations using the 74S182. It will be used in the accumulator for the RFI project.

I. Discussion and Result

The 4-bit adder is pictured in Fig. 1. The point of interest is the reduction in the number of AND gates accomplished by sharing six Boolean terms between S_1 and C_2. A sequence of these 4-bit adders to add two $4n$-bit numbers in ripple-carry fashion would be rather slow, because each next adder would have to wait for the previous carry.

The 3-bit adder is pictured in Fig. 2. Sixteen of these will add two 48-bit numbers faster than twelve of the 4-bit adders. The increase in speed is achieved by a well known "look ahead" arrangement. Each 3-bit adder puts out a $\bar{P}$ bit and a $\bar{G}$ bit instead of a carry bit. The $\bar{G}$ bit will have value 0 if and only if a carry will be generated. The $\bar{P}$ bit will have value 0 if $\bar{G}$ has value 1 and a carry would be propagated. Using the $\bar{P}$s and $\bar{G}$s from all the 3-bit adders at once, the whole arrangement can get the sum of two 48-bit numbers in only 5 gate-delay times.

The main point of interest in the 3-bit adder is the sharing of AND gates by way of the function F. This makes it possible to implement all the functions desired with the given chip. The other point of interest is that F does not depend on C_0. This means that F will find its final value during the time that C_0 is being computed in the lookahead network. Thus when C_0 arrives all the gates affected by C_0 will find their values in just 1 gate-delay time.

The practical problem was to mechanize a 48-bit digital integrator (or accumulator) whose memory could be read and cleared at the same time that a new datum d_i was added to begin a new integral. The memory type being used was not, itself, clearable. The form of the integrator is as shown in Fig. 3.

The available 74S381 4-bit arithmetic/logical unit (ALU) almost met the requirements. The 74S381 can be controlled so as either to add its two inputs (A and B), or to transmit zero output. The zero output clears the memory, as desired, but the time-coincident incoming datum d_i is lost and does not initialize the next summation.

What was required, then, was an ALU which could add two numbers A and B, or transmit one of them (called "pass A"). No such ALU was available commercially. It seemed, however, that an acceptable ALU could be created by programming a field-programmable logic array (FPLA) such as the 74S330, whose package size and pinout were compatible with the 74S381.

The 74S330 has 50 AND gates, each of which may be programmed to AND any of 12 input variables using arbitrary assignments of which "rail" or true/false sense of each input variable would be effective on each AND gate. Further, any combination of the AND gate outputs can be ORed into any of six internal OR gates. Each OR gate is made available as an output from the 74S330, and each OR output can be programmed so as to invert its output, if desired.

The 48-bit adder logic was to be built on a circuit board having little room ("real estate") for extra components. It was, therefore, necessary to maximize the number of bits which could be added by each 74S330. Further, the adder had to be a "fast" adder, i.e., compatible with carry lookahead techniques, such as those based on the 74S182 lookahead generator. It soon became apparent that the availability of only 50 AND gates in the FPLA was going to be a major limitation in the adder word-size or adder speed or both. Thus a combinatorial study was made in an attempt to obtain an optimal design for an adder based on the 50-gate FPLA.

First choice would have been a 4-bit adder with the $\bar{P}$, $\bar{G}$, and E functions. We think that is impossible although we do not know how to prove it.

The design chosen after several trials was the 3-bit adder of Fig. 4.

Arthur L. Rubin has obtained the following result. Subject to the requirements of the chip 74S330, a 3-bit adder with $\bar{P}$, $\bar{G}$, and E which makes no use of a tie around (such as the connection of the output F to other FPLA inputs, as seen in Fig. 2) from the output of the chip (and hence avoids the one extra gate delay) cannot be designed with fewer than 50 AND gates. Achieving the 3-bit adder of Fig. 4 on one chip makes the 48-bit adder as fast as possible.

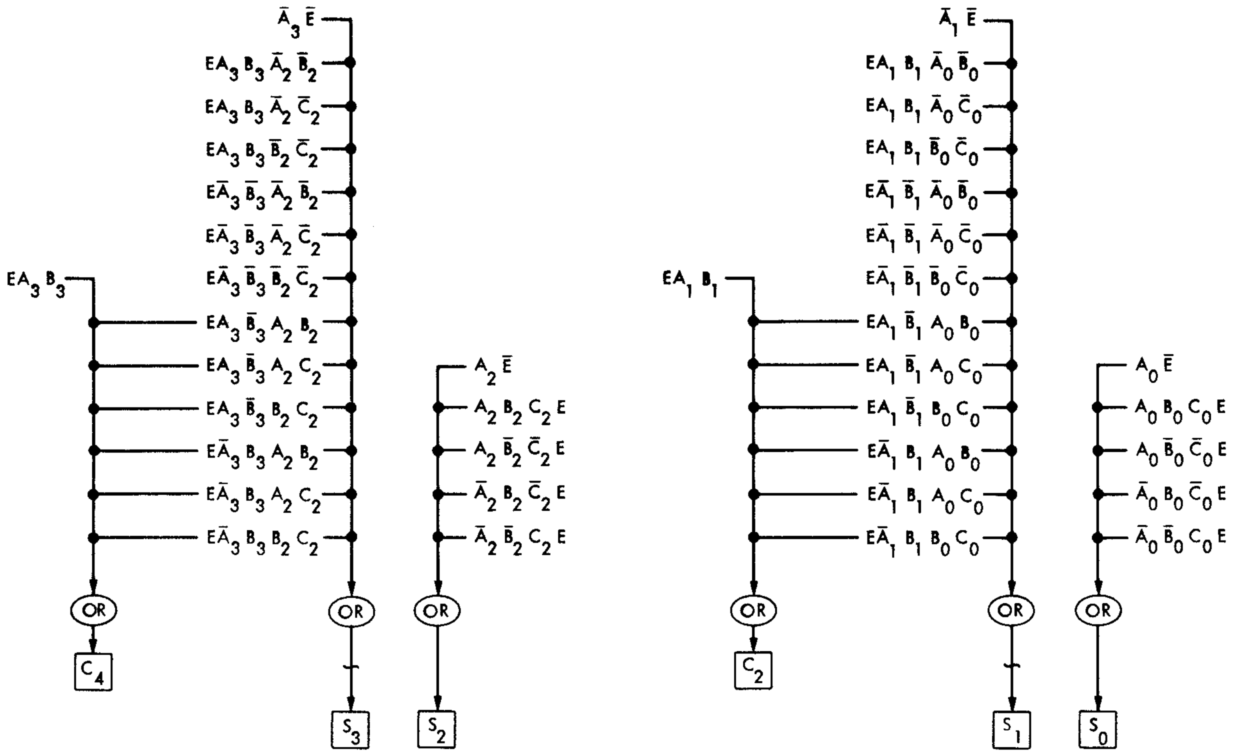
$\bar{A}_3 \bar{E}$
$EA_3 B_3 \bar{A}_2 \bar{B}_2$
$EA_3 B_3 \bar{A}_2 \bar{C}_2$
$EA_3 B_3 \bar{B}_2 \bar{C}_2$
$E\bar{A}_3 \bar{B}_3 \bar{A}_2 \bar{B}_2$
$E\bar{A}_3 \bar{B}_3 \bar{A}_2 \bar{C}_2$
$E\bar{A}_3 \bar{B}_3 \bar{B}_2 \bar{C}_2$
$EA_3 B_3$
$EA_3 \bar{B}_3 A_2 B_2$
$EA_3 \bar{B}_3 A_2 C_2$
$EA_3 \bar{B}_3 B_2 C_2$
$E\bar{A}_3 B_3 A_2 B_2$
$E\bar{A}_3 B_3 A_2 C_2$
$E\bar{A}_3 B_3 B_2 C_2$
$A_2 \bar{E}$
$A_2 B_2 C_2 E$
$A_2 \bar{B}_2 \bar{C}_2 E$
$\bar{A}_2 B_2 \bar{C}_2 E$
$\bar{A}_2 \bar{B}_2 C_2 E$
OR
OR
OR
C_4
S_3
S_2
$\bar{A}_1 \bar{E}$
$EA_1 B_1 \bar{A}_0 \bar{B}_0$
$EA_1 B_1 \bar{A}_0 \bar{C}_0$
$EA_1 B_1 \bar{B}_0 \bar{C}_0$
$E\bar{A}_1 \bar{B}_1 \bar{A}_0 \bar{B}_0$
$E\bar{A}_1 \bar{B}_1 \bar{A}_0 \bar{C}_0$
$E\bar{A}_1 \bar{B}_1 \bar{B}_0 \bar{C}_0$
$EA_1 B_1$
$EA_1 \bar{B}_1 A_0 B_0$
$EA_1 \bar{B}_1 A_0 C_0$
$EA_1 \bar{B}_1 B_0 C_0$
$E\bar{A}_1 B_1 A_0 B_0$
$E\bar{A}_1 B_1 A_0 C_0$
$E\bar{A}_1 B_1 B_0 C_0$
$A_0 \bar{E}$
$A_0 B_0 C_0 E$
$A_0 \bar{B}_0 \bar{C}_0 E$
$\bar{A}_0 B_0 \bar{C}_0 E$
$\bar{A}_0 \bar{B}_0 C_0 E$
OR
OR
OR
C_2
S_1
S_0

Fig. 1. 4-bit (dual 2-bit) adder, with enable

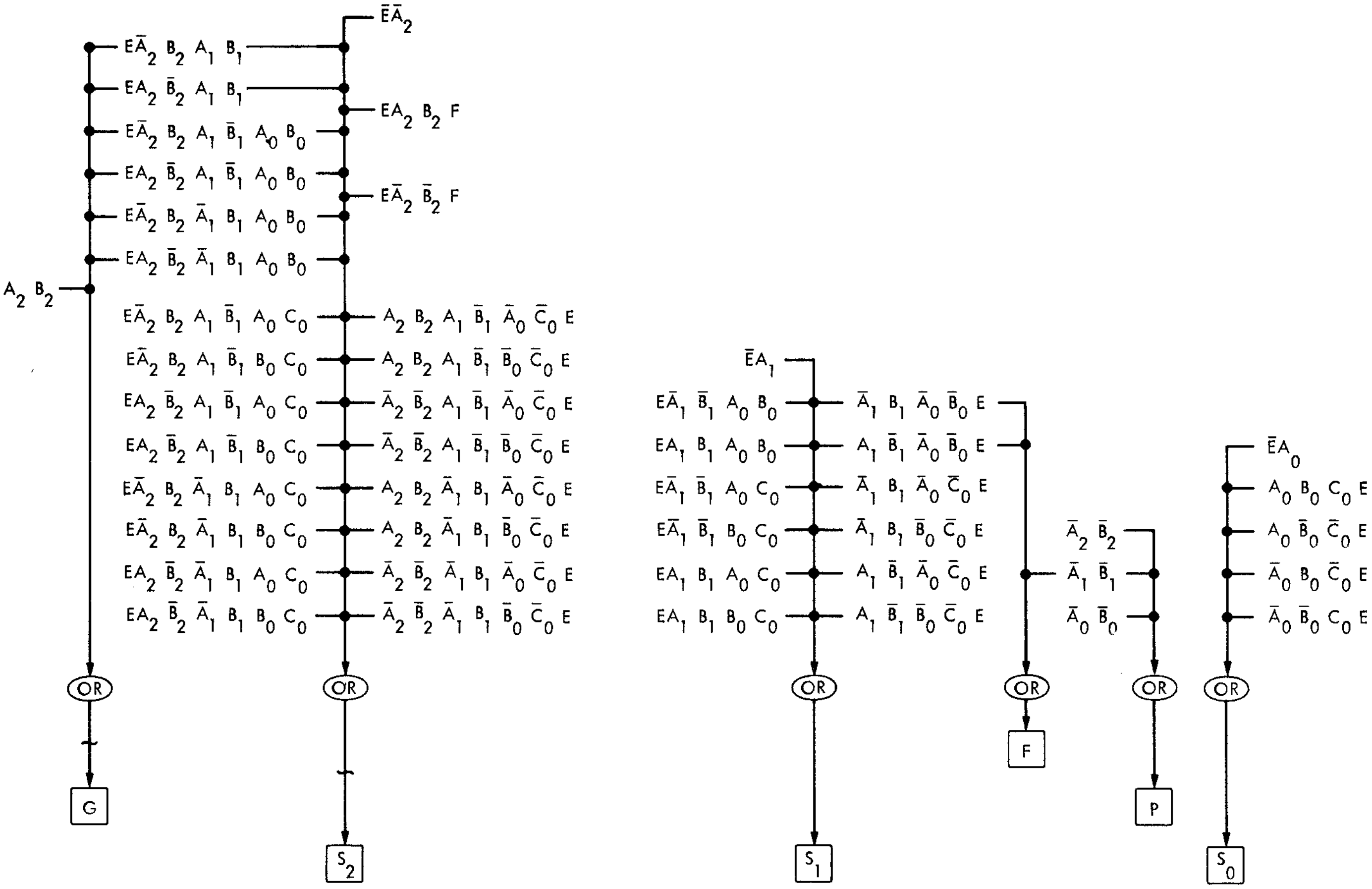
$\bar{E}\bar{A}_2$
$E\bar{A}_2\ B_2\ A_1\ B_1$
$EA_2\ \bar{B}_2\ A_1\ B_1$
$EA_2\ B_2\ F$
$E\bar{A}_2\ B_2\ A_1\ \bar{B}_1\ A_0\ B_0$
$EA_2\ \bar{B}_2\ A_1\ \bar{B}_1\ A_0\ B_0$
$E\bar{A}_2\ \bar{B}_2\ F$
$E\bar{A}_2\ B_2\ \bar{A}_1\ B_1\ A_0\ B_0$
$EA_2\ \bar{B}_2\ \bar{A}_1\ B_1\ A_0\ B_0$
$A_2\ B_2$
$E\bar{A}_2\ B_2\ A_1\ \bar{B}_1\ A_0\ C_0$
$A_2\ B_2\ A_1\ \bar{B}_1\ \bar{A}_0\ \bar{C}_0\ E$
$E\bar{A}_2\ B_2\ A_1\ \bar{B}_1\ B_0\ C_0$
$A_2\ B_2\ A_1\ \bar{B}_1\ \bar{B}_0\ \bar{C}_0\ E$
$EA_2\ \bar{B}_2\ A_1\ \bar{B}_1\ A_0\ C_0$
$\bar{A}_2\ \bar{B}_2\ A_1\ \bar{B}_1\ \bar{A}_0\ \bar{C}_0\ E$
$EA_2\ \bar{B}_2\ A_1\ \bar{B}_1\ B_0\ C_0$
$\bar{A}_2\ \bar{B}_2\ A_1\ \bar{B}_1\ \bar{B}_0\ \bar{C}_0\ E$
$E\bar{A}_2\ B_2\ \bar{A}_1\ B_1\ A_0\ C_0$
$A_2\ B_2\ \bar{A}_1\ B_1\ \bar{A}_0\ \bar{C}_0\ E$
$E\bar{A}_2\ B_2\ \bar{A}_1\ B_1\ B_0\ C_0$
$A_2\ B_2\ \bar{A}_1\ B_1\ \bar{B}_0\ \bar{C}_0\ E$
$EA_2\ \bar{B}_2\ \bar{A}_1\ B_1\ A_0\ C_0$
$\bar{A}_2\ \bar{B}_2\ \bar{A}_1\ B_1\ \bar{A}_0\ \bar{C}_0\ E$
$EA_2\ \bar{B}_2\ \bar{A}_1\ B_1\ B_0\ C_0$
$\bar{A}_2\ \bar{B}_2\ \bar{A}_1\ B_1\ \bar{B}_0\ \bar{C}_0\ E$
OR
OR
G
S_2
$\bar{E}A_1$
$E\bar{A}_1\ \bar{B}_1\ A_0\ B_0$
$\bar{A}_1\ B_1\ \bar{A}_0\ \bar{B}_0\ E$
$EA_1\ B_1\ A_0\ B_0$
$A_1\ \bar{B}_1\ \bar{A}_0\ \bar{B}_0\ E$
$E\bar{A}_1\ \bar{B}_1\ A_0\ C_0$
$\bar{A}_1\ B_1\ \bar{A}_0\ \bar{C}_0\ E$
$E\bar{A}_1\ \bar{B}_1\ B_0\ C_0$
$\bar{A}_1\ B_1\ \bar{B}_0\ \bar{C}_0\ E$
$EA_1\ B_1\ A_0\ C_0$
$A_1\ \bar{B}_1\ \bar{A}_0\ \bar{C}_0\ E$
$EA_1\ B_1\ B_0\ C_0$
$A_1\ \bar{B}_1\ \bar{B}_0\ \bar{C}_0\ E$
$\bar{A}_2\ \bar{B}_2$
$\bar{A}_1\ \bar{B}_1$
$\bar{A}_0\ \bar{B}_0$
$\bar{E}A_0$
$A_0\ B_0\ C_0\ E$
$A_0\ \bar{B}_0\ \bar{C}_0\ E$
$\bar{A}_0\ B_0\ \bar{C}_0\ E$
$\bar{A}_0\ \bar{B}_0\ C_0\ E$
OR
OR
OR
OR
S_1
F
P
S_0

Fig. 2. 3-bit, lookahead-compatible adder, with enable, using an intermediate output F connected to higher-order sum-terms

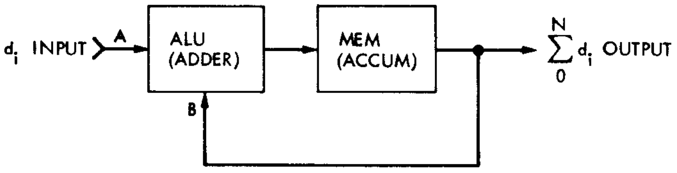

Fig. 3. Schematic diagram of digital accumulator using an adder

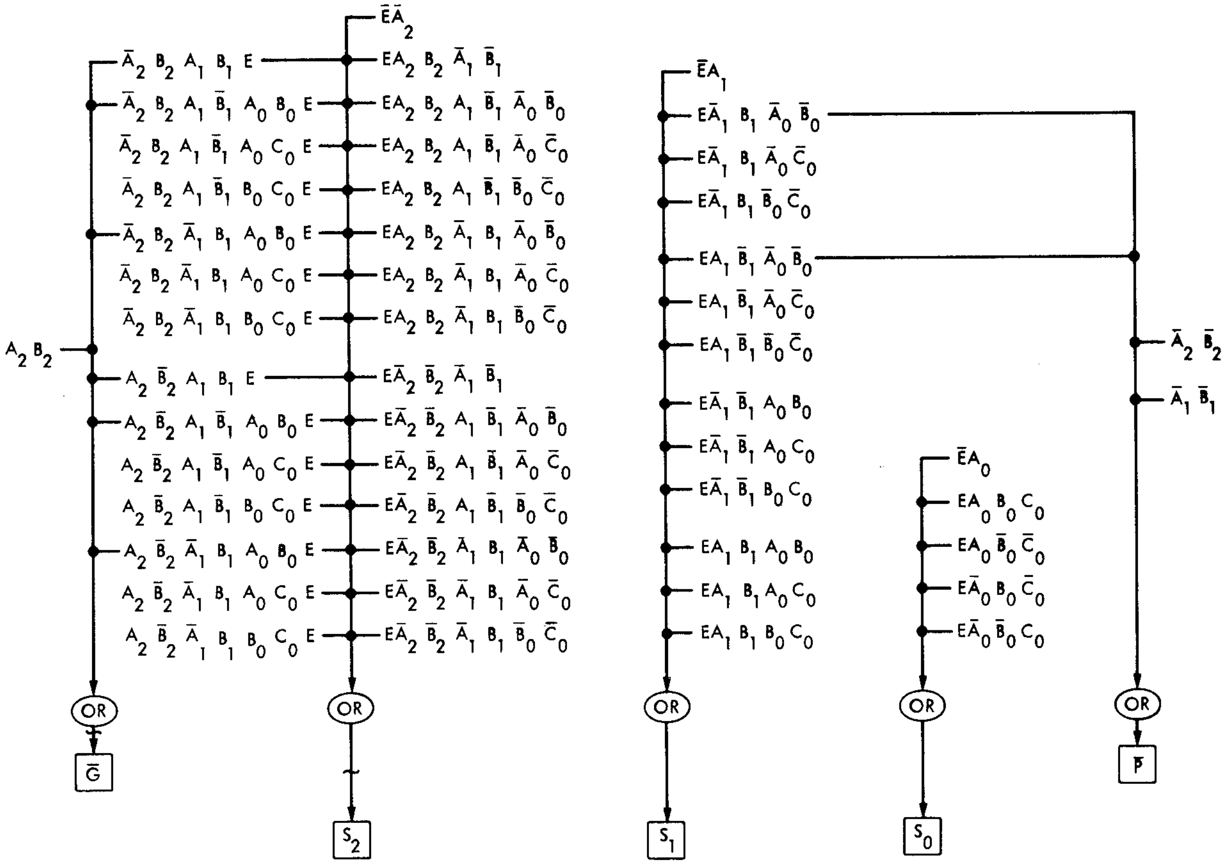
$\bar{E}\bar{A}_2$
$\bar{A}_2 B_2 A_1 B_1 E$
$EA_2 B_2 \bar{A}_1 \bar{B}_1$
$\bar{A}_2 B_2 A_1 \bar{B}_1 A_0 B_0 E$
$EA_2 B_2 A_1 \bar{B}_1 \bar{A}_0 \bar{B}_0$
$\bar{A}_2 B_2 A_1 \bar{B}_1 A_0 C_0 E$
$EA_2 B_2 A_1 \bar{B}_1 \bar{A}_0 \bar{C}_0$
$\bar{A}_2 B_2 A_1 \bar{B}_1 B_0 C_0 E$
$EA_2 B_2 A_1 \bar{B}_1 \bar{B}_0 \bar{C}_0$
$\bar{A}_2 B_2 \bar{A}_1 B_1 A_0 B_0 E$
$EA_2 B_2 \bar{A}_1 B_1 \bar{A}_0 \bar{B}_0$
$\bar{A}_2 B_2 \bar{A}_1 B_1 A_0 C_0 E$
$EA_2 B_2 \bar{A}_1 B_1 \bar{A}_0 \bar{C}_0$
$\bar{A}_2 B_2 \bar{A}_1 B_1 B_0 C_0 E$
$EA_2 B_2 \bar{A}_1 B_1 \bar{B}_0 \bar{C}_0$
$A_2 B_2$
$A_2 \bar{B}_2 A_1 B_1 E$
$E\bar{A}_2 \bar{B}_2 \bar{A}_1 \bar{B}_1$
$A_2 \bar{B}_2 A_1 \bar{B}_1 A_0 B_0 E$
$E\bar{A}_2 \bar{B}_2 A_1 \bar{B}_1 \bar{A}_0 \bar{B}_0$
$A_2 \bar{B}_2 A_1 \bar{B}_1 A_0 C_0 E$
$E\bar{A}_2 \bar{B}_2 A_1 \bar{B}_1 \bar{A}_0 \bar{C}_0$
$A_2 \bar{B}_2 A_1 \bar{B}_1 B_0 C_0 E$
$E\bar{A}_2 \bar{B}_2 A_1 \bar{B}_1 \bar{B}_0 \bar{C}_0$
$A_2 \bar{B}_2 \bar{A}_1 B_1 A_0 B_0 E$
$E\bar{A}_2 \bar{B}_2 \bar{A}_1 B_1 \bar{A}_0 \bar{B}_0$
$A_2 \bar{B}_2 \bar{A}_1 B_1 A_0 C_0 E$
$E\bar{A}_2 \bar{B}_2 \bar{A}_1 B_1 \bar{A}_0 \bar{C}_0$
$A_2 \bar{B}_2 \bar{A}_1 B_1 B_0 C_0 E$
$E\bar{A}_2 \bar{B}_2 \bar{A}_1 B_1 \bar{B}_0 \bar{C}_0$
OR
$\bar{G}$
OR
S_2
$\bar{E}A_1$
$E\bar{A}_1 B_1 \bar{A}_0 \bar{B}_0$
$E\bar{A}_1 B_1 \bar{A}_0 \bar{C}_0$
$E\bar{A}_1 B_1 \bar{B}_0 \bar{C}_0$
$EA_1 \bar{B}_1 \bar{A}_0 \bar{B}_0$
$EA_1 \bar{B}_1 \bar{A}_0 \bar{C}_0$
$EA_1 \bar{B}_1 \bar{B}_0 \bar{C}_0$
$E\bar{A}_1 \bar{B}_1 A_0 B_0$
$E\bar{A}_1 \bar{B}_1 A_0 C_0$
$E\bar{A}_1 \bar{B}_1 B_0 C_0$
$EA_1 B_1 A_0 B_0$
$EA_1 B_1 A_0 C_0$
$EA_1 B_1 B_0 C_0$
OR
S_1
$\bar{E}A_0$
$EA_0 B_0 C_0$
$EA_0 \bar{B}_0 \bar{C}_0$
$E\bar{A}_0 B_0 \bar{C}_0$
$E\bar{A}_0 \bar{B}_0 C_0$
OR
S_0
$\bar{A}_2 \bar{B}_2$
$\bar{A}_1 \bar{B}_1$
OR
$\bar{P}$

Fig. 4. 3-bit, fastest lookahead-compatible adder, with enable, using no intermediate outputs